Application of Machine Learning on Software Quality Assurance and Testing: A Chronological Survey

Mohammad Hossain*
University of Minnesota Crookston, MN, USA

Hongkai Chen†
University of California San Diego, CA, USA

Abstract

Ensuring the quality is essential for a successful Software System. Software systems need to be tested in every stage of the Software Development Life Cycle (SDLC) irrespective of the type of software being developed. If a software bug remains undetected in the early phase of the SDLC, it becomes harder to fix it at a later stage and becomes very costly. The application of machine learning in Software Quality Assurance and Testing can help testers in the testing process, including the early detection and prediction of a software bug. However, employing machine learning techniques brings new challenges to testing and quality assurance. Machine Learning (ML) uses Artificial Intelligence (AI) techniques that focus on a given dataset to find any trend present in the data. It has been observed that some software testing activities can, in fact, be represented as a learning problem. Thus, ML can be used as an efficient tool to automate software-testing activities, especially when the software system becomes very complex. This survey aims to study and summarize the application of machine learning on software quality assurance and testing in a chronological manner by selecting from articles published in the last twenty-six years or so.

Key Words: Software quality assurance and testing, machine learning, artificial intelligence, chronological survey, neural network, support vector machine.

1 Introduction

Machine learning (ML) is the study of computer algorithms designed to exhibit intelligence by self-learning through experience observed from its surrounding environment. Machine learning is a branch of artificial intelligence. The basic idea of machine learning is to build a model based on sample data or training data to predict or make decisions without programming. Today, machine learning is widely used in many industries and applications, including pattern recognition, computer vision, aeronautical engineering, finance, entertainment, computational biology, biomedical engineering, and medical applications [3].

Software quality assurance and testing refer to testing the software and ensuring the proper quality of the software. It is a crucial phase of SDLC in terms of time and money. Much effort has been taken to reduce the cost of the testing phase to keep the software development cost within the budget. For that, machine learning has been introduced in software testing for a long time [12, 45]. However, the use of machine learning brings some new challenges to the quality assurance and testing field. At the same time, it also provides some potential new methods for software testing. This paper will discuss the application of machine learning in software quality assurance and testing.

In the following section, we discussed our methodology for this study. The next section summarizes our findings in various subsections as found in different year groups. Finally, the following section analyzed our findings on various machine learning techniques.

2 Methodology

For this survey, we collected fifty different papers published from 1995 to 2021. We tried to investigate the trend that might be found in utilizing artificial intelligence in general and machine learning in particular in software testing and quality assurance over the period of the last twenty-six years or so. We grouped the papers into five-year periods and tried to focus on the main discussion topics of each group. Table 1 shows the papers that we gathered for this purpose.

3 Summaries of Our Findings

This section listed the papers that we studied in chronological order in the following subsections. We tried to group the papers uniformly in a five-year span with the exception of 3.5 and 3.6 to keep the number of papers in each group somewhat uniform. We discussed the authors' motivation for their research and their findings in each subgroup.

3.1 1995-1999

In this period, we have studied two papers. In [3], the authors talked about the problem of a large volume of test cases generated by automated tools as the effectiveness of these test cases is not clear. The authors present experimental results on using a neural network for pruning a test case set while preserving its effectiveness. The authors concluded that based

* hossain@crk.umn.edu.
† hongkai@ucsd.edu.

Table 1: Articles sorted by year with key words

Year	Number of Articles	Keywords	References
1995	2	Neural Network (NN)	[3], [12]
2002	1	Neural Network	[45]
2003	1	Software Engineering	[50]
2004	1	Active Learning, Automatic Classification, Markov models	[6]
2006	3	Infeasible Paths, Data Flow Based Testing, Classification, Value-Based Software, Test Data Generation, Decision Tree, Genetic Algorithms	[11], [46], [49]
2007	2	Support Vector Machines, MartiRank, Test Oracle, Statistical Software Testing	[5], [32]
2008	4	Decision trees, Fault-proneness prediction, Test Oracles Generation, NN, Support Vector Machines, Defect-prone Software Modules	[8], [7], [19], [25]
2009	1	Category-Partition, Black Box Testing	[9]
2010	2	Bayesian Reasoning, Asymmetric Function, NN	[35], [41]
2011	4	Classification Framework, Metamorphic Testing, GUI Testing, Test Oracle, Support Vector Machines, Grammar Induction, Clustering, Regression Test	[13], [20], [36], [48]
2012	5	Black Box Testing, Clustering, GUI Testing, NN, Test Oracle, Mutation Testing	[1], [21], [22], [32], [43]
2013	3	Test Coverage Criteria, Combining Testing Techniques, Data Mining	[14], [28], [47]
2016	1	Metamorphic Relations, Graph Kernels	[26]
2017	3	Aging Related Bug, Data Mining, Big Data, Metamorphic Testing	[31], [32], [33]
2018	4	Test Oracle, Dataset Diversity, Metamorphic Testing	[34], [35], [36], [37]
2019	8	Test Case Generation, Classification, Clustering, Test Automation, NN, Fault Localization	[38], [39], [40], [41], [42], [43], [44], [45]
2020	4	Data Cleaning, NN, Code Review, Continuous Integration	[46], [47], [48], [49]
2021	1	Statistical Regression	[50]

on their experiment, neural networks are promising test case effectiveness predictors. They further concluded that their method is able to adapt as the software matures with sufficient accuracy. In [12], the authors tried to accurately estimate the cost of software testing. Authors applied machine learning techniques to determine the software testing attributes that are important in predicting software testing costs and time.

Figure 1 shows the distribution of the articles over the years.

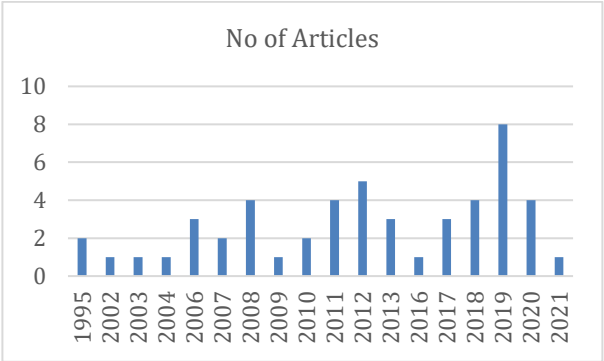

Figure 1: Distribution of the articles over the years

3.2 2000-2004

In this period, we selected three papers [6, 45, 50] to discuss. In [45], the authors talked about a test oracle to determine whether a given test case exposes a fault or not. This paper presents a new concept of using a two-layered artificial neural network as
an automated oracle for testing a real software system. The authors concluded that the neural network is a promising method of testing a software application capable of learning new versions of evolving software. In [50], the authors discussed the domain of the machine learning approach and how it can be utilized in software engineering. They showed how the software development and maintenance tasks could be formulated as learning problems. In [6], the authors focused on the automatic classification of program behavior using execution data. They introduced a technique that models program executions as Markov models and also devised a clustering method for Markov models to combine multiple program executions to form an effective behavior classifier.

3.3 2005-2009

We found ten papers [5, 8, 9, 11, 17, 19, 25, 32, 46, 49] to study in this period. In [46], the authors spent on infeasible paths, basically on three main approaches: prediction, classification, and identification of infeasibility. They also addressed these aspects in the scope of integration and object-oriented testing. The authors claimed that the finding of this paper would aid in the planning of the testing activity and in the establishment of testing strategies. In [49], the authors proposed a framework for value-based software test data generation. Value-based software engineering considers value into the software engineering principles and practices as opposed to value-neutral software engineering where each product in software development, such as requirement, use case, test case, and defect, is treated as equally important. The authors talked about applying machine learning methods to value-based software engineering in this paper. In [11], the authors analyzed

the problems in software code and proposed a model that will help catch those problems earlier in the project life cycle using machine learning methods. In [32], authors discussed issues of testing machine learning applications and proposed machine learning algorithms such as SVM and MurtiRank. In [5], the authors presented an adaptive sampling mechanism using machine learning for software testing. In [8], the author provided a brief overview of state of art and reports on a number of novel applications of machine learning in the area of software testing. In [19], the author argued that there is no general technique for estimating software fault-proneness. In this paper, the author proposed the use of machine learning for software fault-proneness prediction. In [25], the authors talked about the artificial neural network for test oracles generation, the same theme as seen in [45]. In [17], the authors evaluated the capability of SVM in predicting defect-prone software modules and claimed that the prediction performance of SVM is generally better than the models they compared with. In [9], the authors proposed a methodology and a tool based on machine learning to help people understand the limitations of test suites and their possible redundancies so that people are able to refine them in a cost-effective manner. The authors claimed that their proposed solution to show promising results on a case study involving students as testers.

3.4 2010-2014

In this period, we found fourteen papers [1, 13-14, 20-22, 28, 32, 35-36, 41, 43, 47-48] to study. In [35], the authors provided a brief overview of some popular Bayesian reasoning methods for achieving reliable and efficient software testing and program analysis. They also explained why those Bayesian reasoning methods are applicable to software testing. In [41], the authors focused on the application of machine learning tools and variable selection tools in order to solve the problem of estimating the execution effort of functional tests. The authors concluded that there is a high complexity of the effort estimation problem. Managers and other testing professionals require easy-to-use and efficient estimation methods. In [36], the authors introduced a classification framework that can help to systematically review research work in the ML and software testing domains. This framework can be used to construct guidelines for choosing the most appropriate learning method and then using it in the software testing stage. The authors claimed that the classification framework is quite strong in capturing various aspects of work in ML and software testing. Furthermore, it helps researchers systematically investigate and extract the prominent information from existing research works in ML and software testing. In [48], the authors presented a technique based on "metamorphic testing" for testing the implementations of machine learning classification algorithms. The authors claimed that their approach enables users and programmers to easily and effectively verify and validate the machine learning components of the software. In [20] and [21], the authors talked about GUI software testing. They proposed avoiding infeasible test cases altogether by predicting which test cases are infeasible using two supervised machine learning methods: support vector machines (SVMs) and grammar induction. The authors concluded that classifying test case feasibility is possible. In [13], the authors introduced a semi-supervised clustering method named semi-supervised K-means (SSKM) to improve cluster test selection. The authors claimed that the semi-supervised clustering method SSKM could improve test selection in most cases. In [32], the authors discussed software quality improvement by early prediction of error patterns. They advocated the use of case-based reasoning (i.e., CBR) to build a software quality prediction system with the help of human experts. In [1], the authors compared the use of artificial neural networks (ANN) and info-fuzzy networks (IFN) as automated oracles to confirm that the developed software complies with its specification and determine whether a given test case exposes faults. The authors concluded that IFN outperforms the ANN for faults causing a large number of faulty records, while the ANN appears to be more suitable for identifying hard-to-detect faults in more stable versions. The IFN clearly outperforms the ANN with respect to training time. In [43], the authors proposed an approach to classifying mutants as a tool to reduce the number of mutants to be executed and to evaluate the quality of test suits without executing them against all possible mutants. They concluded that the results obtained so far are encouraging, but this approach still needs more experiments to fully confirm its validity. In [22], the authors discussed some of the relationships between the work of Artificial Intelligence (AI) techniques and Software Engineering (SE) problems. In [28], the authors proposed an approach using machine learning techniques to link test results from the application of different testing techniques. The authors claimed that the major advantage of this approach is that it can automatically determine equivalence classes, which are, in general, manually determined according to subjective rules. In [14], the authors presented a method that can combine testing techniques adaptively during the testing process. The authors concluded that this method could improve the fault detection effectiveness with respect to single testing techniques and their random combination. In [47], the authors presented cooperative testing and analysis, including human-tool cooperation and human-human cooperation. The authors claimed that this could reduce human efforts and burden in software engineering activities.

3.5 2015-2018

In this period, we found eight papers [15, 18, 24, 26-27, 30, 33-34] to study. In [26], authors conducted a feature analysis to identify the most effective features for predicting metamorphic relations for testing scientific software using graph kernels. In [27], the authors presented a study on the application of machine learning techniques and static source code metrics to predict aging-related bugs. The authors concluded that static source code metrics could be used as predictors for aging-related bugs. In [18], the authors discussed many different works in the field of software vulnerability analysis and discovery that utilize machine-learning and data-mining techniques. In [15], the authors presented a framework for validating the large-scale

image data as well as adequately verifying both the software tools and machine learning algorithms. They claimed that the framework addresses the most important issues of verification and validation in big data. In [30], the authors discussed problems with ML applications and discovered software engineering approaches and software testing research areas to solve these problems. They concluded some key areas such as deep learning, fault localization, and prediction. In [24], the authors discussed the characteristics of some machine learning algorithms and concluded the main challenges of testing machine learning applications. Meanwhile, the authors presented two techniques: comparing results of different implementations and metamorphic testing to mitigate the test oracle problem. In [33], the author reviewed two traditional views of service and product qualities of a machine learning software. In [34], the authors demonstrated a new metamorphic testing method that can be used to test neural network learning models. This method mainly lies on dataset diversity and behavioral oracle. The authors conjectured that their approach could be effective in the software testing of machine learning programs.

3.6 2019-2021

In this period, we selected thirteen papers to discuss [2, 7, 10, 16, 23, 29, 31, 38-40, 42, 44, 51]. In [31], the authors proposed a self-adaptive learning-based test framework that learns the optimal policy of generating stress test cases for different types of software systems. The paper [39] presented a strategy to identify the tests that have passed or failed by combining clustering and semi-supervised learning. The paper [23] discussed that the Artificial Intelligence key pillars that can be used in software testing and talked about how the future will look like in terms of artificial intelligence and software testing. In [40], authors used a deep neural network to build a software defect prediction model and compared their proposed model with other machine learning algorithms like random forests, decision trees, and naive Bayesian networks. The authors claimed that their results showed small improvement over the other learning models in most cases. In [2], the authors proposed a methodology predict and localize faults in a software system. The authors used the random forest machine learning technique to train their model. The articles [38] and [44] talked about the use of machine learning in software quality assurance and prediction. In the article [16], authors studied 48 different papers focusing on making a survey of research efforts based on using ML algorithms to support software testing. The authors believed their mapping study would provide significant insights into machine learning applied to software testing. In [7], the authors discussed the current existing testing practices for ML programs. And the authors also explained the main sources of faults in an ML program. In [10], the authors focused on the relevant features of a large dataset in order to improve the accuracy of software quality estimation. The authors concluded that machine learning algorithms could help to estimate the quality level of software. In [51], the authors presented a comprehensive survey of machine learning testing research. The authors summarized the current research status of different ML testing properties, testing components, and testing workflow. In [42], the authors presented their work that can reduce the need for manual reviews by automatically identifying which code fragments should be reviewed manually. The authors concluded that their work could improve the speed of code reviews. In [29], the authors investigated the application for STEP of five machine learning (ML) models reported as the most accurate ones when applied to SDLC effort prediction.

4 Analysis Based on Various ML Techniques

This section analyzed various ML techniques used in software testing, which we study in this survey. Table 2 summarizes our findings.

Table 2: ML techniques used for software testing in this survey

Techniques	References	Number of Articles
Neural Network	[1], [3], [8], [11], [19], [25], [34], [40], [41], [42], [44], [45], [51]	13
Support Vector Machine	[8], [17], [19] [20], [21], [32], [41], [43]	8
Clustering	[1], [13], [39]	3
Decision Tree	[8], [10], [11], [28], [38], [40]	6
Grammar Induction	[20], [21]	2
Bayesian Based Method	[14], [24], [35], [38], [40]	5
Random Forest	[2], [10], [38], [40]	4
Generic ML Techniques	[5], [7], [9], [12], [15], [16], [18], [22], [23], [26], [27], [29], [30], [31], [32], [33], [36], [46], [47], [48], [49], [50]	22

4.1 Neural Network

Neural network is widely used in software testing. We found thirteen articles talking about this technique. Among the articles, neural network is used for fault-proneness prediction in [19] and is used for automatic test oracles generation in [25] and [1]. In [41], neural network is used to estimate the execution effort of software testing. In [34], neural network is used for metamorphic testing. In [40], neural network is used for software defect prediction. In [42], neural network is used for identifying code fragments for manual review.

4.2 Support Vector Machines

Support vector machines is widely used in software testing. We found eight articles talking about this technique. Among the articles, support vector machines is used for generating reliable test oracle in [32]. In [19] and [17], a support vector

machine is used for fault-proneness prediction. In [41], the support vector machine is used to estimate the execution effort of software testing. In [20] and [21], the support vector machine is used for GUI testing. In [43], support vector machines is used for mutation testing.

4.3 Clustering, Decision Tree, and Grammar Induction

We found three articles talking about the clustering technique. In [13], clustering is used for improving regression test selection. Clustering is used for automatic test oracles generation in [1] and is used for classifying test outcomes in [39].

We found six articles talking about the decision tree. Decision tree is used for defect prediction in [8] and [11]. In [40], decision tree is used for software defect prediction. In [10] and [38], decision tree is used for software quality prediction.

We found two articles talking about grammar induction. In both [20] and [21], grammar induction is used for GUI testing.

4.4 Bayesian Based Method and Random Forest

We found five articles talking about Bayesian based method. In [14], Bayesian based method is used for combining testing techniques. In [40], Bayesian based method is used for software defect prediction. In [43], Bayesian based method is used for software quality prediction.

We found four articles talking about random forest. In [40], random forest is used for software defect prediction. In [2], random forest is used for software fault localization. In [10] and [38], random forest is used for software quality prediction.

5 Software Testing Activities Found in this Study

In our study we found different types of software testing activities that uses different types of machine learning techniques. We are discussing few of them in the following subsections.

5.1 Creating Test Data

In [49], the authors proposed a framework for value-based software test data generation. Value-based software engineering considers value into the software engineering principles and practices as opposed to value-neutral software engineering where each product in software development, such as requirement, use case, test case, and defect, is treated as equally important. The authors talked about applying machine learning methods to value-based software engineering in this paper.

5.2 Test Case Generation

In [31], the authors proposed a self-adaptive learning-based test framework that learns the optimal policy of generating stress test cases for different types of software systems. In test case generation, one of the important tasks is to reduce manual review of the code segments. In [42], the authors presented how machine learning can be used to reduce the need for manual reviews by automatically identifying which code fragments should be reviewed manually.

5.3 User Interface Testing

The success of a software product largely depends on an error free user interface. That is why user interface testing places a vital role in software quality. In [20] and [21], the authors talked about GUI software testing. They proposed avoiding infeasible test cases altogether by predicting which test cases are infeasible using two supervised machine learning methods: support vector machines (SVMs) and grammar induction.

5.4 Regression Testing

A regression testing is done whenever a change has been made to a software product to ensure that the change made did not introduce any new error. For a large product, running an effective regression test is challenging because of the amount of test cases needed to run. In [13], the authors introduced a semi-supervised clustering method named semi-supervised K-means (SSKM) for improving regression test selection. The authors claimed that the semi-supervised clustering method SSKM could improve test selection in most cases.

6 Conclusion

Software testing has been a great area of research as it is a vital issue for producing quality software [4]. Researchers are interested in performing successful testing with minimal effort possible by doing test automation. Machine learning can play an important role in this regard. It is evident from our study that the intersection of these fields has drawn attention from many researchers for a long time. From our study, we have seen many early techniques of machine learning, such as neural network, decision tree, etc., and modern techniques like deep learning are equally applicable in software testing.

Our citations in the paper are arranged in chronological order. A smaller reference number indicates an earlier publication, and a larger number indicates a recent publication. The summary of our study shown in Table 2 shows the popular techniques of ML as applied in software testing. For instance, neural network has been a popular technique all through our selected period. Also, we see that SVM and grammar induction are the techniques found in the middle period, whereas the recent trends are focused on the techniques like decision tree, Bayesian-based method, and random forest. Of course, a substantial number of papers applied machine learning in general without being specific to any particular techniques.

The papers in our study discussed many different approaches to applying ML in software testing and introduced the challenges in them. Researchers have proposed some potential ways to solve those challenges and suggested a number of future directions. The common theme we found in those works is that machine learning techniques can be employed during the software testing process. Using machine learning techniques

can assist testers in predicting software defects, localizing software faults, finding some specific bugs, and improve effectiveness and efficiency. We hope the researchers and practitioners in this field will be benefited from our study.

We have also studied the techniques of the machine learning that can be used for different types of software testing activities such as creating test data, test case generation, testing user interfaces, and regression testing.

In the future, we will explore how machine learning is used for improving the testing of different categories of the software such as object-oriented software, distributed software, etc.

References

[1] D. Agarwal, D. E. Tamir, M. Last, and A. Kandel, "A Comparative Study of Artificial Neural Networks and Info-Fuzzy Networks as Automated Oracles in Software Testing," IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 42(5):1183-1193, 2012.

[2] H. Ali and T. A. Khan, "On Fault Localization Using Machine Learning Techniques," 2019 International Conference on Frontiers of Information Technology (FIT), pp. 357-3575, doi: 10.1109/FIT47737.2019.00073, 2019.

[3] C. Anderson, A. Von Mayrhauser, and R. Mraz, "On the Use of Neural Networks to Guide Software Testing Activities," *Proceedings of 1995 IEEE International Test Conference (ITC)*, IEEE, pp. 720-729, October 1995.

[4] A. Bandi and P. Heeler, "Usability Testing: A Software Engineering Perspective," 2013 International Conference on Human Computer Interactions (ICHCI), pp. 1-8, doi: 10.1109/ICHCI-IEEE.2013.6887809, 2013.

[5] N. Baskiotis, M. Sebag, M. C. Gaudel, and S. D. Gouraud, A Machine Learning Approach for Statistical Software Testing," *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI),* pp. 2274-2279, January 2007.

[6] J. F. Bowring, J. M. Rehg, and M. J. Harrold, "Active Learning for Automatic Classification of Software Behavior," *ACM SIGSOFT Software Engineering Notes*, 29(4):195-205, 2004.

[7] H. B. Braiek. and F. Khomh, "On Testing Machine Learning Programs," *Journal of Systems and Software*, 164:110542, 2020.

[8] L. C. Briand, "Novel Applications of Machine Learning in Software Testing," 2008 The Eighth International Conference on Quality Software, IEEE, pp. 3-10, August 2008,

[9] L. C. Briand, Y. Labiche, Z. Bawar, and N. T. Spido, "Using Machine Learning to Refine Category-Partition Test Specifications and Test Suites," Information and Software Technology, 51(11):1551-1564, 2009.

[10] A. A. Ceran and Ö. Ö. Tanriover, "An Experimental Study for Software Quality Prediction with Machine Learning Methods," 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), IEEE, 2020.

[11] E. Ceylan, F. O. Kutlubay and A. B. Bener, "Software Defect Identification Using Machine Learning Techniques," 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), pp. 240-247, 2006, doi: 10.1109/EUROMICRO.2006.56.

[12] T. J. Cheatham, J. P. Yoo, and N. J. Wahl, "Software Testing: A Machine Learning Experiment," *Proceedings of the 1995 ACM 23rd Annual Conference on Computer Scienc*e, pp. 135-141, February 1995.

[13] S. Chen, Z. Chen, Z. Zhao, B. Xu, and Y. Feng, "Using Semi-Supervised Clustering to Improve Regression Test Selection Techniques," 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation, IEEE, pp. 1-10, March 2011.

[14] D. Cotroneo, R. Pietrantuono, and S. Russo, "A Learning-Based Method for Combining Testing Techniques," 2013 35th International Conference on Software Engineering (ICSE) , IEEE, pp. 142-151, May 2013.

[15] J. Ding, X.-H. Hu, and V. Gudivada. "A Machine Learning Based Framework for Verification and Validation of Massive Scale Image Data," IEEE Transactions on Big Data, 2017.

[16] V. H. S. Durelli, R. Durelli, S. Borges, A. Endo, M. Elder, D. Dias, and M. Guimaraes, "Machine Learning Applied To Software Testing: A Systematic Mapping Study," IEEE Transactions on Reliability, 68.3:1189-1212, 2019.

[17] K. O. Elish and M. O. Elish, "Predicting Defect-Prone Software Modules Using Support Vector Machines," *Journal of Systems and Software*, 81(5):649-660, 2008.

[18] S. M. Ghaffarian, and H. R. Shahriari. "Software Vulnerability Analysis and Discovery Using Machine-Learning and Data-Mining Techniques: A Survey." ACM Computing Surveys (CSUR) 50.4:1-36, 2017.

[19] I. Gondra, "Applying Machine Learning to Software Fault-Proneness Prediction," *Journal of Systems and Software*, 81(2):186-195, 2008.

[20] R. Gove and J. Faytong, "Identifying Infeasible GUI Test Cases Using Support Vector Machines and Induced Grammars," 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, IEEE, pp. 202-211, March 2011.

[21] R. Gove and J. Faytong, "Machine Learning and Event-Based Software Testing: Classifiers for Identifying Infeasible GUI Event Sequences," *Advances in Computers*, Elsevier, 86:109-135, 2012.

[22] M. Harman, "The Role of Artificial Intelligence in Software Engineering," 2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE), IEEE pp. 1-6, June 2012.

[23] H. Hourani, A. Hammad, and M. Lafi, "The Impact of Artificial Intelligence on Software Testing," 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), pp. 565-570, doi: 10.1109/JEEIT.2019.8717439, 2019.

[24] S. Huang, E. H. Liu, Z. W. Hui, S. Q. Tang, and S. J. Zhang, "Challenges of Testing Machine Learning

Applications," *International Journal of Performability Engineering*, 14(6):1275, 2018.

[25] H. Jin, Y. Wang, N. W. Chen, Z. J. Gou, and S. Wang, "Artificial Neural Network for Automatic Test Oracles Generation," 2008 International Conference on Computer Science and Software Engineering, IEEE, 2:727-730, December 2008.

[26] U. Kanewala, J. M. Bieman, and A. Ben ‐ Hur, "Predicting Metamorphic Relations For Testing Scientific Software: A Machine Learning Approach Using Graph Kernels," Software Testing, Verification and Reliability, 26(3):245-269, 2016.

[27] L. Kumar and A. Sureka, "Aging Related Bug Prediction using Extreme Learning Machines," 2017 14th IEEE India Council International Conference (INDICON), pp. 1-6, 2017, doi: 10.1109/INDICON.2017.8487925, 2017.

[28] A. R. Lenz, A Pozo, and S. R. Vergilio, "Linking Software Testing Results with a Machine Learning Approach," Engineering Applications of Artificial Intelligence, 26(5-6):1631-1640, 2013.

[29] C. López-Martín, "Machine Learning Techniques for Software Testing Effort Prediction," *Software Quality Journal*, 30(1):65-100, 2022, https://doi.org/10.1007/s11219-020-09545-8, 2022.

[30] S. Masuda, K. Ono, T. Yasue, and N. Hosokawa,. A survey of software quality for machine learning applications. In 2018 IEEE International conference on software testing, verification and validation workshops (ICSTW), IEEE, pp. 279-284, April 2018.

[31] M. H. Moghadam, M. Saadatmand, M. Borg, M. Bohlin, and B. Lisper, "Machine Learning to Guide Performance Testing: An Autonomous Test Framework," 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), IEEE, pp. 164-167, April 201.

[32] C. Murphy, G. E. Kaiser, and M. Arias, "An Approach to Software Testing of Machine Learning Applications," Proceedings of the 19th International Conference on Software Engineering & Knowledge Engineering, Technical Program, Hyatt Harborside Hotel, Boston, Massachusetts, USA, July 9-11, January 2007.

[33] S. Nakajima, "Quality Assurance of Machine Learning Software," 2018 IEEE 7th Global Conference on Consumer Electronics (GCCE), IEEE, pp. 601-604, October 2018.

[34] S. Nakajima, "Dataset Diversity for Metamorphic Testing of Machine Learning Software," International Workshop on Structured Object-Oriented Formal Language and Method, Springer, Cham, pp. 21-38, November 2018.

[35] A. S. Namin and M. Sridharan, "Bayesian Reasoning for Software Testing," *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, pp. 349-354, November 2010.

[36] M. Noorian, E. Bagheri, and W. Du, "Machine Learning-based Software Testing: Towards a Classification Framework," In SEKE, pp. 225-229, July, 2011.

[37] E. Rashid, P. Srikanta, and V. Bhattacherjee. "A Survey in the Area of Machine Learning and Its Application for Software Quality Prediction," ACM SIGSOFT Software Engineering Notes, 37.5:1-7, 2012.

[38] S. Reddivari and J. Raman. "Software Quality Prediction: an Investigation Based on Machine Learning," 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI), IEEE, 2019.

[39] M. Roper, Using Machine Learning to Classify Test Outcomes," 2019 IEEE International Conference on Artificial Intelligence Testing (AITest), IEEE, pp. 99-100, April 2019.

[40] M. Samir, M. El-Ramly and A. Kamel, "Investigating the Use of Deep Neural Networks for Software Defect Prediction," 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), pp. 1-6, doi: 10.1109/AICCSA47632.2019.9035240, 2019.

[41] D. G. e Silva, M. Jino, and B. T. de Abreu, "Machine Learning Methods and Asymmetric Cost Function to Estimate Execution Effort of Software Testing," 2010 Third International Conference on Software Testing, Verification and Validation, IEEE, pp. 275-284, April 2010.

[42] M. Staron and O. Soder, "Using Machine Learning to Identify Code Fragments for Manual Review," 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, 2020.

[43] J. Strug and B. Strug, "Machine Learning Approach in Mutation Testing," IFIP International Conference on Testing Software and Systems, Springer, Berlin, Heidelberg, pp. 200-214, November 2012.

[44] L. Surya, "Machine Learning-Future of Quality Assurance," *International Journal of Emerging Technologies and Innovative Researc*h (www. jetir. org), ISSN:2349-5162, 2019.

[45] M. Vanmali, M. Last, and A. Kandel, "Using a Neural Network in the Software Testing Process," *International Journal of Intelligent Systems*, 17(1):45-62, 2002.

[46] S. R. Vergilio, J. C. Maldonado, and M. Jino," Infeasible Paths in the Context of Data Flow Based Testing Criteria: Identification, Classification, and Prediction," *Journal of the Brazilian Computer Society*, 12(1):73-88, 2006.

[47] T. Xie, "The Synergy of Human and Artificial Intelligence in Software Engineering," 2013 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), IEEE, pp. 4-6, May 2013.

[48] X. Xie, J. W. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and Validating Machine Learning Classifiers by Metamorphic Testing," *Journal of Systems and Software*, 84(4):544-558, 2011.

[49] D. Zhang, "Machine Learning in Value-Based Software Test Data Generation," 2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06), IEEE, pp. 732-736, November 2006..

[50] D. Zhang and J. J.Tsai, "Machine Learning and Software Engineering," *Software Quality Journal*, 11(2):87-119, 2003.

[51] J. M. Zhang, M Harman, L. Ma, and Y. Liu, , "Machine

Learning Testing: Survey, Landscapes and Horizons," IEEE Transactions on Software Engineering, 2020.

Mohammad Hossain is an Assistant Professor of Software Engineering and IT Management at the University of Minnesota Crookston where he teaches various software engineering courses. He earned his Ph.D. from North Dakota State University in 2016. His Ph.D. dissertation title was "Foundational Algorithms Underlying Horizontal Processing of Vertically Structured Big Data Using pTrees". His research interest includes Data Mining, Machine Learning, Software Engineering, Cybersecurity, Algorithm etc. He is a member of ISCA and served as the Program Chair of CATA 2021 and CATA 2022.

Hongkai Chen received his BS in Software Engineering from University of Minnesota Crookston in 2021. He was a honor student at UMC and received UROP scholarship from University of Minnesota. He is currently pursuing his MS in Computer Science at University of California, San Diego. His research interest includes Software Engineering, Machine Learning, Cybersecurity, etc.